CLAIMS

1-5. (Canceled)

6. (Currently amended) A data management system, the[[said]] system characterized as a composite system comprising at least one computer processor, and the system comprising a plurality of processes;

each process having an interface and implementing at least one respective service defined by that interface;

a first invocation of the at least one respective service by a transaction resulting in the creation of a first transaction local to the process thereof,

the first local transaction being a child of the invoking transaction and being parent of any transaction triggered by invocation of a service of another process;

a second invocation of the at least one respective service by a transaction resulting in the creation of a second transaction local to the process thereof,

the second local transaction being a child of the invoking transaction and being parent of any transaction triggered by invocation of a service of another process;

each process characterized in that if the first transaction and the second transaction conflict but are both children of a same invoking transaction, then the first transaction and the second transaction are not executed concurrently;

each process further characterized in that each transaction local thereto is independently handled at the process; and

each process making scheduling and recovery decisions independent of any centralized component.

7. (Currently amended) The system of claim 6 wherein a root transaction is able to dynamically set concurrency preferences for a resulting distributed transaction, based on client needs,

wherein the concurrency preferences specify the extent to which shared resource access is desired or allowed or denied among descendant transaction invocations of the root invocation or user, and

based on other, concurrent transaction invocations [[who]]that are also descendants of the same root.

8. (Currently amended) A method for use with a data management system, the[[said]] system characterized as a composite system comprising at least one processor, and the system comprising a plurality of processes;

each process having an interface and implementing at least one respective transactional service defined by that interface[[;]],

wherein invocation of the at least one respective transactional service is by a thread of the invoking transaction, and

wherein each process is[[being]] a parent of any transaction triggered by invocation of a transactional service of another process;

each process further characterized in that each transaction local thereto is independently handled at the process;

each process making scheduling and recovery decisions independent of any centralized component triggered by invocation of a transactional service of another process[[,]]; and

~~each process further characterized in that each transaction local thereto is independently handled at the process,~~

each process making scheduling and recovery decisions independent of any centralized component, the method comprising the steps of:

propagating from a first process to a second process a message indicative of a globalCommit operation with respect to a root transaction,

wherein said message is also indicative of a number or identifying list of transactional invocations [[which]]that the first process has made to the second process on behalf of the root transaction;

within the second process, comparing the number or list indicated in the message with a count or list within the second process of the number or list of transactional invocations [[which]]that have been made on behalf of the root transaction; and

in the event the comparison yields a non-match, aborting the transaction.

9. (Previously presented) The method of claim 8 wherein each process is executed using

Java.

10. (Currently amended) A method for use with a data management system, the[[said]] system characterized as a composite system, and the system comprising a plurality of processes[[,]];

each process having an interface and implementing at least one respective service defined by that interface,

wherein invocation of the at least one respective service by a transaction results[[ing]] in the creation of a transaction local to the process thereof, and

wherein the local transaction is[[being]] a child of the invoking transaction and is[[being]] a parent of any transaction triggered by invocation of a service of another process[[,]];

each process further characterized in that each transaction local thereto is independently handled at the process[[,]];

each process making scheduling and recovery decisions independent of any centralized component, the method comprising the steps of:

propagating from a first process to a second process a message indicative of a globalCommit operation with respect to a root transaction,

wherein said message is also indicative of a number or list of invocations [[which]]that the first process has made to the second process on behalf of the root transaction;

within the second process, comparing the number or list indicated in the message with a count or list within the second process of the number or list of invocations [[which]]that have been made on behalf of the root transaction;

in the event the comparison yields a match, proceeding with the globalCommit operation.

11. (Currently amended) A method for use with a data management system, said system characterized as a composite system, and the system comprising a plurality of processes[[,]];

each process having an interface and implementing at least one respective service defined

by that interface,

wherein invocation of the at least one respective service by a transaction results[[ing]] in the creation of a transaction local to the process thereof, and

wherein the local transaction is[[being]] a child of the invoking transaction and is[[being]] a parent of any transaction triggered by invocation of a service of another process;

each process further characterized in that each transaction local thereto is independently handled at the process[[,]];

each process making scheduling and recovery decisions independent of any centralized component, the method comprising the steps of:

propagating from a first process to a second process a message indicative of a globalCommit operation with respect to a root transaction,

wherein said message is also indicative of a number or list of invocations which the first process has made to the second process on behalf of the root transaction;

within the second process, comparing the number or list indicated in the message with a count or list within the second process of the number or list of invocations [[which]]that have been made on behalf of the root transaction;

in the event the comparison yields a non-match, aborting the transaction.

12. (Currently amended) A distributed transaction system ~~using a two-phase commit protocol~~, said system characterized as a composite system comprising at least one computer processor, the system comprising:

a plurality of processes[[;]], each process having an interface and implementing at least one respective service defined by that interface[[;]],

wherein each service invocation provides application-level comments about the nature of the invocation to the service, or

wherein any globalCommit requires a registration, and the registration for a globalCommit also supplies such additional application-level comments at the time of registering for globalCommit, or

wherein each or any globalCommit~~two-phase commit~~ message exchange between processes carries application-level comments~~also carrying information about the actual work being committed~~.

13. (Currently amended) The system of claim 12, such information being logged for recoverability in the event of a crash, such information being used for assistance at any time before, during, or after global commitment.

14. (Canceled)

15. (Currently amended) A method for use in a distributed transaction system, said system characterized as a composite system, the system comprising:

a plurality of processes, each process having an interface and implementing at least one respective service defined by that interface, the method comprising the steps of:

by each service invocation, providing application-level comments about the nature of the invocation to the service; or

propagating a registration for a globalCommit,

wherein the registration for a globalCommit also supplies such additional application-level comments; or

exchanging[[for]] each globalCommit message exchanged between processes,

wherein each globalCommit message includes application-level comments~~including also information about the actual work being committed~~.

16. (Currently amended) The method of claim 15 further comprising the step of logging such information for recoverability in the event of a crash, such information being used for assistance at any time before, during, or after global commitment.

17. (Canceled)

18. (Currently amended) A distributed system, the[[said]] system characterized as a composite system comprising at least one computer processor, and the system comprising a plurality of processes;

each process having an interface and implementing at least one respective transactional

service defined by that interface[[;]],

wherein a root transactional invocation or, alternatively, a root's human user, is allowed to dynamically set its or the user's[[his]] concurrency preferences for an entire transactional invocation[[;]],

wherein the root invocation (transaction) propagates the concurrency preferences with each or any child invocation [[it]] the root invocation makes[[;]], and

wherein the propagated concurrency preferences at any level in the root invocation's invocation hierarchy specify the extent to which shared resource access is desired or allowed or denied among descendant transactional invocations of the root invocation or user and other, concurrent transactional invocations [[who]]that are also descendants of the same root.

19. (Canceled)

20. (Previously presented) The system of claim 18 wherein any or each invocation propagates the concurrency preferences as it has received them from the root invocation.

21-22. (Canceled)

23. (Currently amended) A computerized data management system, referred to as transactional service, wherein the transactional service is executed on a local server, comprising:

one or more operations that can be invoked by remote clients,

wherein the remote clients [[being]]are executed on a remote server that is distinct from the local server[[;]], and

wherein some or all such remote clients have[[ing]] one or more associated transaction contexts;

an invocation of the service, by a remote client, also containing partial or complete information that indicates[[ing]] or contains[[ing]] said client's transaction context or contexts;

an invocation of the service, by a remote client, of an operation leading to a new transaction different from, but possibly related to, any existing client transaction[[;]],

wherein such an operation-level transaction [[being]]is committed before the client

transaction context is terminated before globalCommit notification;

the transactional service locally, on the local server, maintaining an undo operation for such a committed operation; and

a failing or failed remote client transaction context leading to the execution of the locally-maintained undo operations of the corresponding committed invocations in the transactional service.

24. (Previously presented) The system of claim 23 where some or all undo operations are executed in an order that is the reverse of an order of their original counterparts.

25. (Previously presented) The system of claim 23 where in addition the undo operations are chosen or defined in the same system as the one where corresponding normal operations were executed.

26. (Previously presented) The system of claim 23 where some or all undo operations are unknown to the transactional context of a remote client.

27. (Currently amended) The system of claim 23 where some or all undo operations are executed after a timeout and are independent of whether the client's transaction context outcome requires such undo.

28. (Currently amended) The system of claim 23 wherein an undo operation is with respect to an original operation, and

wherein an undo operation's effects are confined to data managed by the transactional service on which the undo operation is maintained, even if the respective original operation involved other services.

29. (Previously presented) The system of claim 23 where the service keeps locks on transactions to ensure that undo operations can be executed correctly.

30. (Original) The system of claim 23 where client context-related information is also part of any global commit message exchanges.

31. (Original) The system of claim 23 where client context information includes application-specific data.

32. (Currently amended) The system of claim 31 where all or part of the context information is logged by storing on persistent storage, and is retrievable by a human administrator.

33. (Original) The system of claim 23 where the service accepts messages indicative of which previously committed operations have to be undone.

34. (Previously presented) The system of claim 23 where the service accepts messages indicative of which previously committed operations do not have to be undone.

35. (Previously presented) The system of claim 23 where some or all invocations are message-based or asynchronous.

36. (Original) The system of claim 23 where some or all invocations are synchronous.

37. (Previously presented) The system of claim 23 where the client can request the undo executions of its invocations at the service while still allowing globalCommit.